来尝试下EF下的数据库事务锁吧

**模板代码**

认真看下面的代码，代码开启了3个线程，其中，线程2最先进入事务，1、3随后进入

System.Data.IsolationLevel.ReadUncommitted：事务的级别

static void Main(string[] args)

{

Task.Run(() =>

{

MyContext myContext = new MyContext();

Thread.Sleep(1000);

using (var tran = myContext.Database.BeginTransaction(System.Data.IsolationLevel.ReadUncommitted))

{

Console.WriteLine("线程1已进入事务");

var entity = myContext.Entity1s.FirstOrDefault(e => e.Code == "Entity1\_1");

Console.WriteLine("线程1已加载实体");

entity.Name = entity.Name + "A";

myContext.SaveChanges();

Console.WriteLine("线程1更改数据了，但是没提交");

// 休眠 20s

Thread.Sleep(20000);

Console.WriteLine("线程1提交了");

tran.Commit();

}

});

Task.Run(() =>

{

MyContext myContext = new MyContext();

using (var tran = myContext.Database.BeginTransaction(System.Data.IsolationLevel.ReadUncommitted))

{

Console.WriteLine("线程2已进入事务");

Thread.Sleep(5000);

var entity = myContext.Entity1s.FirstOrDefault(e => e.Code == "Entity1\_1");

Console.WriteLine("线程2已加载实体");

entity.Name = entity.Name + "A";

myContext.SaveChanges();

Console.WriteLine("线程2更改数据了，但是没提交");

Console.WriteLine("线程2提交了");

tran.Commit();

}

});

Task.Run(() =>

{

// 休眠 5s，好让线程1先进入事务

Thread.Sleep(5000);

MyContext myContext = new MyContext();

var entity = myContext.Entity1s.FirstOrDefault(e => e.Code == "Entity1\_1");

Console.WriteLine("线程3已加载实体");

});

Console.ReadKey();

}

**ReadUncommitted（读未提交）级别输出**

线程2已进入事务

线程1已进入事务

线程1已加载实体

线程1更改数据了，但是没提交

线程2已加载实体

线程1提交了

线程2更改数据了，但是没提交

线程2提交了

线程3已加载实体

结论：当线程1SaveChanges后，这条数据就被锁住了，线程2无法SaveChanges，需要等线程1Commit才行，线程3没有使用事务，所以线程3查询时需要等等线程1、2Commit才行

**ReadCommitted（读已提交）级别输出**

线程2已进入事务

线程1已进入事务

线程1已加载实体

线程1更改数据了，但是没提交

线程1提交了

线程2已加载实体

线程2更改数据了，但是没提交

线程2提交了

线程3已加载实体

结论：当线程1SaveChanges后，这条数据就被锁住了，线程2无法读取这条数据，需要等线程1Commit才行，同理线程3